NVIDIA AUGSTAS VEIKTSPĒJAS RISINĀJUMA *CUDA* PIELIETOŠANA ATTĒLU APSTRĀDEI

NVIDIA HIGH PERFORMANCE COMPUTING SOLUTION CUDA APPLICATION FOR IMAGE PROCESSING

Autors: **Jānis STANKEVIČS**, e-mail: janis.stankevics@gmail.com, telefons: 26739913
Zinātniskā darba vadītājs: **Sergejs Kodors, Dr.sc.comp.,** e-mail: sergejs.kodors@rta.lv
Rēzeknes Tehnoloģiju akadēmija,
Atbrīvošanas aleja 115, Rēzekne, Latvija

Abstract. *The goal of this research report is to compare the program execution times using the Central Processing Unit and NVIDIA's high throughput solution CUDA to execute the program on the Graphic Processing Unit. The programs will run an algorithm to convert a raster image into dotted half-tone. CUDA`s programming model allows programmers to run their programs on a parallel GPU architecture consisting of several blocks of threads that are being ran in parallel. This dynamic technology allows to further decrease the time it takes to run a program, by simply adding more GPU's or upgrading the existing one*

The research experiment concluded that NVIDIA solution is faster than using usual CPU serial code.

Keywords: *CUDA, high throughput, Graphic Processing Unit, execution time.*

Ievads

Kopš 2005. gada centrālā procesora darbības frekvences uzlabojumi ir samazinājušies, un ātrums ir palicis apmēram tāds pats ar katru jauno procesoru paudzi. Viens no risinājumiem ir palielināt paralēlisma iespējas. Tāpēc mūsdienās *CPU* ražotāji vairāk koncentrējas uz paralēlisma palielināšanu, piemēram, vairāku kodolu procesora un hiperpavedienu tehnoloģijas.

Tikmēr video kartes ražotājs *NVIDIA* 2006. gadā izlaida *CUDA* rīkkopu, kas atļauj programmētājiem izmantot *NVIDIA GPU* kodolus vairāku, vienkāršu uzdevumu vienlaicīgai izspildīšanai, kas ļauj samazināt slodzi uz centrālā procesora un paātrināt programmas izspildes laiku. Bet šī arhitektūra nespēj pilnībā aizstāt centrālā procesora uzdevumus un ir daudz lēnāka izspildot vienkāršu sērijveida kodus, tāpēc šim darbam joprojām tiek izmantots centrālais procesors.

Šī darba **mērķis** ir salīdzināt programmas izspildes laiku starp centrālo procesoru un CUDA risinājumu.

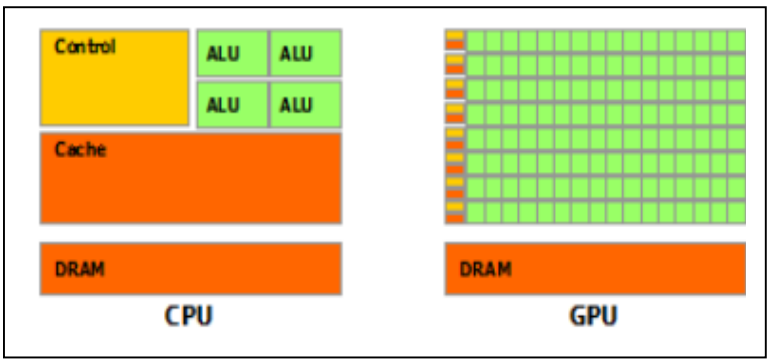
Lai sasniegtu mērķi tika izvirzīti šadi **uzdevumi:**

1) Literatūras analīze par *CUDA* tehnoloģiju.
2) Realizēt programmu, kas pārveido attēlu no rastra grafikas uz melnbaltu rastra iespiedumu (*half-tone*) vektorgrafiku pielietojot centrālo procesoru un *CUDA* tehnoloģiju.
3) Realizēt programmu, kas pārveido attēlu no rastra grafikas uz krāsainu rastra iespiedumu (*color half-tone*) vektorgrafiku, pielietojot centrālo procesoru un *CUDA* tehnoloģiju.
4) Salīdzināt attēla apstrādes laikus.

CUDA tehnoloģija

Viena no video kartes galvenajiem uzdevumiem ir liela daudzuma parallēlskaitļošana priekš grafiskās renderēšanas, tāpēc vairāk vietas tiek veltīts skaitļošanas kodoliem un mazāk kešatmiņām un plūsmas vadībām. Videokartēm, salīdzinot ar centrālajiem procesoriem, ir daudz vairāk kodolu, kas ir mazāk efektīvāki izspildīt instrukcijas pēc iespējas ātrāk, bet lielais skaits kodolu ir paredzēti liela daudzuma sarežģītām operācijām ar peldošo komatu izspildei.

http://dx.doi.org/10.17770/het2018.22.3634



1.attēls. **Procesora un videokartes arhitektūra** [1]

Lai būtu iespējams izmantot *NVIDIA* videokartes priekšrocības vispārēju uzdevumu skaitļošanas uz grafiskās kartes (*General-purpose computing on graphics processing units - GPGPU*) veikšanai ir nepieciešams instalēt papildus rīkkopu *CUDA toolkit*. Šī rīkkopa satur speciālo *NVIDIA* izstrādāto C/C++ kompilatoru, kas ļauj centrālajam procesoram dot komandas priekš *GPU*, lai izspildītu uzdevumus.

CUDA paralēlprogrammēšanas modelis paredz to, ka matemātiskais uzdevums tiks sadalīts vairākās grupās kas sastāv no pavedieniem(*threads)*,katra no grupām satur līdz 1024 pavedieniem. Katrs pavedienies veiks kodola(*kernel*) funckcijas izspildi(piem. katrs pavedienis ir *for* cikla iterācija). Katrs pavediena bloks ir neatkarīgs no cita, tas nozīmē, ka *CUDA* programmas ātrdarbība ir iespējams palielināt vienkārši palielinot skaitļošanas kodolu daudzumu.

Materiāli un metodes

Darba metode: eksperimentālā metode.

Eksperimenta apraksts: šajā darba ietvaros tiks veikta eksperimentālā daļa kur attēlu apstrādei tiks izmantota rastra iespiedums*(half-tones)* paņemiems, kurš pārveido rastra grafikas pikseļus par dažāda izmēra apaļiem vektora "punktiem". Kā apstrādajamais attēls priekš pētnieciskā daļas tika izvēlēts attēls "Lena". Tiks uzņemts paša attēla apstrādes laiks, neieskaitot atmiņas izdales un informācijas nolasīšanas/ierakstīšanas laikus, jo tie atšķiras starp abām metodēm. Tiks izmantoti šāda attēla izmēri: 512x512, 1024x1024, 2048x2048, 4096x4096.

Melnbalta attēla apstrāde ar rastra iespieduma metodi

Priekš melnbalta attēla apstrādes melna krāsas punkta diametra izmērs ir atkarīgs no blakus esošā pikseļu melnās krāsas intensitātes. Melnbalts attēls tika parveidots no *.pgm formāta uz *.svg formātu. Autors izvēlējās tieši *PGM* formātu, jo attēla informācija tiek saglabāta kā ascii simboli un to ir vienkārši izmantot, un nav nepieciešamība izmantot papildus bibliotēkās. 2. attēlā tiek aprakstīts algoritms pseidokodā.

1. Tiek nolasīta informācija:
   Attēla izmēri tiek saglabāti mainīgajos **height** un **width**.Attēla pikseļa vērtības tiek saglabātas masīvā **Image** ar *int* datu tipu.
2. Tiek fiksēts sākuma laiks izmantojot:
   -CPU- *clock_t*
   -CUDA- *cudaEvent_t*
3. Attēla asptrāde pēc šāda algoritma:
   counter ← 0
   Visiem **h** no 0 līdz **height** ar soli 2 izspildīt:
   Visiem **w** no 0 līdz **width** ar soli 2 izspildīt:
   $$\boldsymbol{r\_buff_{counter}} = 1 - (\mathbf{Image_{w\ h}} + \mathbf{Image_{w+1\ h}} + \mathbf{Image_{w\ h+1}} + \mathbf{Image_{w+1\ h+1}})/\ \mathbf{1020}$$
   $\boldsymbol{h\_buff_{counter}}$=h+1;
   $\boldsymbol{w\_buff_{counter}}$=r+1;
   Palielināt **counter** par 1
   Iekšējā cikla beigas
   Ārējā cikla beigas
   kur:
   h- Attēla vertikālās kordinātes
   w- Attēla horizontālās kordinātes
4. Tiek fiksēts beigu laiks. Izspildes laiks tiek izvadīts uz ekrāna
5. Tiek izmantota struktūra lai ierakstītu datus pēc **.svg* formātā
6. Programmas beigas

2.attēls. **Melnbalta attēla apstrādes algoritms**

Krāsaina attēla apstrāde ar rastra iespieduma metodi

Kā avota datnes formāts priekš krāsaina attēla tika izvēlēts *.bmp, jo var viegli piekļūt un apstrādāt noteiktos pikseļus. Tika izmantota *EasyBMP* bibliotēka lai saglabātu visu attēla

Priekš krāsaina attēla punkta diametrs paliek vienmērigs, bet tā krāsas noteikšanai tiek izmantots euklīda algoritms, kas ir parādīts 3. attēlā [2].

1. Tiek nolasīts attēls, izmantojot *EasyBMP* bibliotēku:
   Attēla pikseļa krāsas vērtības tiek saglabātas attiecīgajos *int* datu tipa masīvos **Red**(sarkans), **Green**(zaļs), **Blue**(blue).Attēla augstums un platums tiek saglabāti attiecīgi **height** un **width**
2. Tiek fiksēts sākuma laiks izmantojot:
   -CPU- *clock_t*
   -CUDA- *cudaEvent_t*
3. Visiem **h** no 0 līdz **height** ar soli 2 izspildīt:
   Visiem **w** no 0 līdz **width** ar soli 2 izspildīt:
   Tiek atrasts loga vidējās vērtības priekš katras krāsas:
   $$\boldsymbol{X}_{avg} = \frac{\mathbf{X}_{w\ h} + \mathbf{X}_{w+1\ h} + \mathbf{X}_{w\ h+1} + \mathbf{X}_{w+1\ h+1}}{4}$$
   Kur: **X** –Attiecigā krāsa (**Red**,**Green**,**Blue**)
   h- Attēla vertikālās kordinātes
   w- Attēla horizontālās kordinātes
   Katram loga pikselim tiek apreķināta euklīda distanci no vidējām vērtībām
   $$F(X)_i = \sqrt{(\boldsymbol{Red}_{avg} - \boldsymbol{Red}_i)^2 + (\mathbf{Green}_{avg} - \mathbf{Green}_i)^2} + (\mathbf{Blue}_{avg} - \mathbf{Blue}_I)^2$$
   Tiek atrasts pikselis ar mazāko vērtību, tā vērtības tiek irakstītas izvades masīvos
   Iekšējā cikla beigas
   Ārējā cikla beigas
4. Tiek fiksēts beigu laiks. Izspildes laiks tiek izvadīts uz ekrāna
5. Tiek izmantota struktūra lai ierakstītu datus pēc **.svg* formātā
6. Programmas beigas

3.attēls. **Krāsaina attēla apstrādes algoritms**

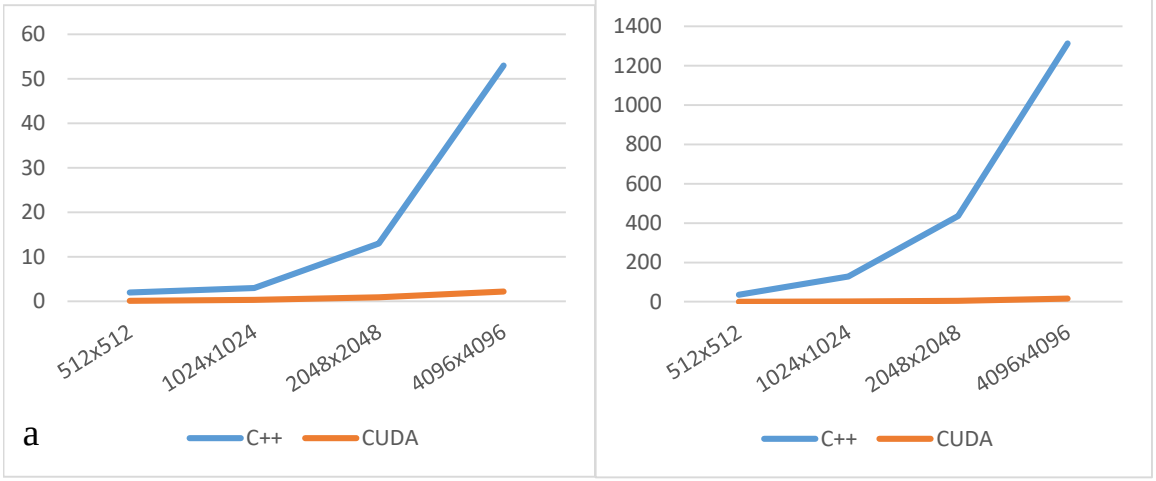
Rezultāti un izvērtējums

Priekš pētījumam tika izmantots dators ar šādu aparatūru:

- Centrālais procesors:*AMD Ryzen 7 1700X Eight-Core Processor (16 CPUs), ~3.4GHz*
- Video karte: *NVIDIA GeForce GTX 1060 6GB*
- Operatīvā atmiņa: *16 GB RAM*

Izmantojamā programmatūra:

- Operatīvā sistēma: *Windows 10 Education 64-bit.*
- Izstrādes vide: *Microsoft Visual Studio 2015 Community*.
- Paralēlskaitļošanas platforma: *CUDA toolkit 8.0*.
- *PGM* un *BMP* attēlu pārskatprogramma: *IrfanView*.
- *SVG* attēlu pārksatprogramma: *Inkscape.*



	Melnbalts attēls (a)		Krāsains attēls (b)	
Attēla izmērs	C++, ms	CUDA, ms	C++, ms	CUDA, ms
512x512	2ms	0.091ms	36ms	0.288ms
1024x1024	3ms	0.288ms	129ms	1.079ms
2048x2048	13ms	0.876ms	437ms	4.2ms
4096x4096	53ms	2.162ms	1314ms	15.6ms

Algoritma izpildes laiks tika salīdzināts starp, izmantojot tika centrālo procesoru un izmantojot *CUDA* risinājumu. Eksperimenta rezultāti parāda to, ka priekš melnbalta attēla *CUDA* risinājums ir 20 reizes ātrāks, bet priekš krāsainiem attēliem tas ir līdz pat 100 reizes ātrāks. Šī atsķirība ir saistīta ar to, ka krāsaina attēla apstrādes algoritms ir sarežģītāks un sastāv no vairākām matemātiskām operācijam.

Secinājumi

Tika veiksmīgi realizētas *CPU* un *CUDA* programmas melnbalta un krāsaina attēla apstrādei ar rastra iespiedumu. Salīdzinot attēla apstrādes laikus tika secināts ka *CUDA* risinājums ir 20 reizes effektīvāks priekš melnbaltiem attēliem un 100 reizes priekš krāsainiem attēliem. Atšķirība *CUDA* risinājuma efektivitātē starp abiem algoritmiem ir tāpēc, ka krāsaina attēla apstrādei ir daudz lielāks izspildāmo matemātisko darbību daudzums.

Tomēr lielākais *CUDA* tehnoloģijas trūkums ir tāds, ka dati ir jāpārsūta pa mātesplates kopni starp operatīvo atmiņu un *GPU* ierīci. Šī darbība var smagi ietekmēt kopējo programmas izspildes laiku. Tāpēc autors uzskata, ka *CUDA* risinājumu ir visslabāk pielietot gadījumos kad ir jāveic liels daudzums matemātisko operacīju uz datu vienību.

Visi darba uzdevumi tika veiksmīgi izspildīti un darba mērķis tika sasniegts.

Summary

Ever since advancements in processing speeds have stalled out, IT companies and researchers have been looking into parallelism to increase computing efficiency. One of such parallelism technologies is general purpose computing on graphics processing unit. One of the GPU manufacturing companies NVIDIA released their solution that allows programmers to write programs that can use GPU resources for computation on software level. Unlike CPU's, GPU architecture is designed to have several, smaller processing cores that are specifically designed for large amount of floating point computations, but at the cost of longer wait times between operations (latency).

For this research report the goal was set to compare the computing times of 2 versions an algorithm, that converts a raster image into half-tone,

A program that uses standard C++ serial code and will run on central processing unit.

A program that will use CUDA and use GPU for large amount of computations.

The results show that using CUDA gives about 20 times faster computation times for greyscale images and even up to 100 times faster for colored images. The large computation efficiency difference between greyscale and colored images is because the algorithm to convert colored images uses more complex mathematical operations than greyscale one.

Literatūras saraksts

1. CUDA C Programming Guide http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#axzz4fNGmi8QR
2. Euclidean Distance – Raw, Normalised, and Double-Scaled Coefficients http://www.pbarrett.net/techpapers/euclid.pdf